SineKAN: Kolmogorov-Arnold Networks using sinusoidal activation functions

Eric Reinhardt*, Dinesh Ramakrishnan and Sergei Gleyzer

Department of Physics and Astronomy, The University of Alabama, Tuscaloosa, AL, United States

Recent work has established an alternative to traditional multi-layer perceptron neural networks in the form of Kolmogorov-Arnold Networks (KAN). The general KAN framework uses learnable activation functions on the edges of the computational graph followed by summation on nodes. The learnable edge activation functions in the original implementation are basis spline functions (B-Spline). Here, we present a model in which learnable grids of B-Spline activation functions are replaced by grids of re-weighted sine functions (SineKAN). We evaluate numerical performance of our model on a benchmark vision task. We show that our model can perform better than or comparable to B-Spline KAN models and an alternative KAN implementation based on periodic cosine and sine functions representing a Fourier Series. Further, we show that SineKAN has numerical accuracy that could scale comparably to dense neural networks (DNNs). Compared to the two baseline KAN models, SineKAN achieves a substantial speed increase at all hidden layer sizes, batch sizes, and depths. Current advantage of DNNs due to hardware and software optimizations are discussed along with theoretical scaling. Additionally, properties of SineKAN compared to other KAN implementations and current limitations are also discussed.

KEYWORDS

machine learning (ML), periodic function, Kolmogorov-Arnold Representation, Kolmogorov-Arnold Networks (KANs), sinusoidal activation function

1 Introduction

Multi-layer perceptrons (MLPs) are a fundamental component of many current leading neural networks (Rumelhart et al., 1986a,b). They are often combined with feature extracting tools, such as convolutional neural networks (LeCun et al., 1989; He et al., 2015; Lecun et al., 1998) and multi-head attention (Vaswani et al., 2017), to create many of the best performing models, such as transformers. One of the key mechanisms that makes MLPs so powerful is that the layers typically end in non-linear activation functions which enables universal approximation from any arbitrary input space to any arbitrary output space using only a single sufficiently wide layer (Hornik et al., 1989). While MLPs enable any such arbitrary mapping, the number of neurons required to achieve that mapping can also be arbitrarily large.

Recent work (Liu et al., 2024) has presented an alternative to the MLP architecture, based on the Kolmogorov-Arnold Representation Theorem (Kolmogorov, 1956, 1957; Braun and Griebel, 2009), accordingly denoted as Kolmogorov-Arnold Networks (KANs) (Liu et al., 2024). In earlier seminal work, Kolmogorov (1956, 1957), it was established that any arbitrary multivariate function can be approximated with a sum of continuous univariate functions over a single variable. In Liu et al. (2024), it was shown that this approximation can be extrapolated to neural network architectures leading to competitive performance with MLPs at often significantly smaller model sizes (Rumelhart et al., 1986a,b). In this work, we will use an efficient implementation of the KAN with learnable B-Spline activation functions (B-SplineKAN) (Cao, 2024) that is numerically consistent with the original implementation of KAN, but on the order of three to five times faster

than the original implementation (Liu et al., 2024) for the purpose of performance comparison.

As Figure 1 shows, the order of operations in traditional MLPs is: on-edge weight multiplication, summation on node, addition of bias, followed by the application of the activation function. In KANs, the order of operations is: learnable activation function on edge, summation on node and optional addition of bias on node. This alternative order of operations satisfies the Kolmogorov-Arnold Representation Theorem and can potentially allow significantly smaller computational graphs compared to MLPs (Liu et al., 2024).

The work establishing KAN as a viable model (Liu et al., 2024) explored the use of B-Splines as the learnable activation function. There is a strong motivation for the choice of B-Splines. Using B-Splines, it is possible to change the size of layer's grid of spline coefficients without meaningfully affecting the model itself, enabling downstream fine-tuning. It is also possible to sparsify the model through a process of pruning low-impact spline terms. It is additionally possible to determine the functional form of the model symbolically. Liu et al. (2024) found that B-Spline models achieved competitive results with MLP layers on a broad range of tasks. The choice of B-Splines isn't without its costs however, as B-SplineKAN layers are significantly slower than MLPs and, while recent implementations have helped to close the gap, MLPs are still substantially faster. Furthermore, there are many tasks presented in Liu et al. (2024) where MLPs still outperformed the B-SplineKAN. Recent work has shown that alternatives to B-SplineKAN can achieve competitive performance under fair comparison (Shukla et al., 2024). In this paper we present SineKAN, a KAN implementation with sine activation functions which aims to address size and speed limitations of common KAN models by replacing B-Spline functions with periodic sine functions. We will also compare to an existing periodic KAN model, the FourierKAN (Xu et al., 2024).

In this work we will introduce the novel SineKAN model functional form and provide empirical evidence that it can achieve comparable performance to B-Spline KAN models and outperform FourierKAN models on some common benchmark tasks. We also show that it can partially avoiding catastrophic forgetting during continual learning, a property which has helped drive interest in other KAN models. In Section 3 we describe the SineKAN architecture, whether it satisfies a universal approximation, and outline a weight initialization strategy that scales consistently with differing grid sizes and stabilizes numerical performance across

multi-layer models. In Section 4, we present results of model inference speed and performance on the MNIST benchmark and compare it with B-SplineKAN and FourierKAN implementations. We discuss our results in Section 5 and summarize our findings in Section 6.

2 Related work

A number of alternative univariate functions to B-Splines have been explored for use in KANs, including wavelets (Bozorgasl and Chen, 2024), Chebyshev polynomials (Sidharth et al., 2024), fractional functions (Aghaei, 2024a), rational Jacobi functions (Aghaei, 2024b), radial basis functions (Ta, 2024), and even variations on Fourier expansions (Xu et al., 2024), discussed in detail in Section 3.2.

Periodic activation functions in neural networks have been explored extensively and shown to provide strong approximations for a broad class of problems. Such problems include general functional modeling (Gallant and White, 1988), image classification (Zhumekenov et al., 2019; wo Wong et al., 2002), and (Sopena et al., 1999; Parascandolo et al., 2016) for general classification tasks. Work using sinusoidal representational networks (Sitzmann et al., 2020) has shown that sinusoidal activation functions lead to strong performance on problems with continuous domains (Lei et al., 2022) and potentially discontinuous domains (Origer and Izzo, 2024; Li et al., 2024). These promising results in sinusoidal activations motivate sine functions as a potentially strong alternative to other explored activation functions for KANs.

3 SineKAN

3.1 Sinusoidal activation function

Here, we propose an alternative to the B-SplineKAN architecture described in Section 1 that is based on sine functions. Mathematically each layer can be expressed as:

$$y_i = \sum_{j,k} A_{ijk} \sin(\omega_k x_j + \phi_{jk}) + b_i \qquad (1)$$

where y_i are the layer output features, x_j are the layer input features, ϕ_{jk} is a phase shift over the grid and input dimensions, ω_k is a grid
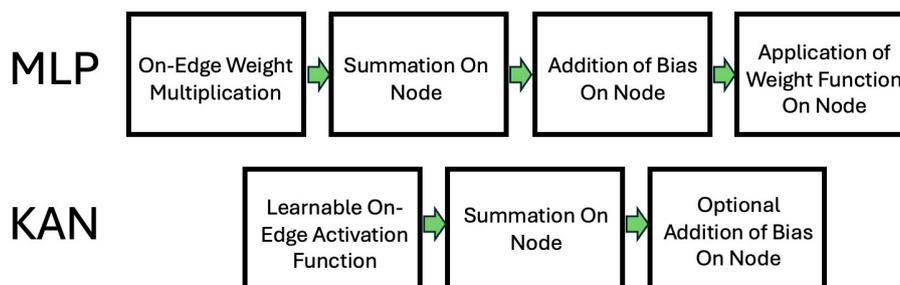

FIGURE 1
Flow of operations. **Top**: MLP. **Bottom**: KAN.

frequency, A_{ijk} are the amplitude weights, and b_i is a bias term. The base functional form of the sines are fixed, while the functional form of the sine activations is learned through learnable frequency and amplitude terms performed over a grid of fixed phases.

3.2 Grid phase shift

In previous work using Fourier series, KAN networks use the form of a full Fourier series expansion, denoted as the following (Xu et al., 2024):

$$y_i = \sum_j \sum_k A_{ijk} \sin(kx_j) + B_{ijk} \cos(kx_j) + b_i \tag{2}$$

where y_i are the layer output features, x_j are the layer input features, A_{ijk} and B_{ijk} are Fourier weight matrices, and b_i is a bias. Here, there is an additional three-dimensional weight matrix compared to SineKAN:

$$y_i = \sum_{j,k} A_{ijk} \sin(\omega_k x_j + \phi_{jk}) + b_i \tag{3}$$

By introducing learnable frequencies ω_k over a grid with fixed phase shifts ϕ_{jk} and input dimensions, we reduce the number of learnable parameters from $O(2oig)$ to $O(oig + g)$ where o is the output dimension, i is in the input dimension, and g is the grid size. We conjecture later in Section 3.4 that this still satisfies universal approximation in the large model limit.

Under the initial assumption for the first layer of the model, a naive approach for initializing the grid weights is to cover a full phase shift range, where the grid phase shift terms would be a range of values from 0 to π. However, it can be shown that, for the following case:

$$\sum_{i=1}^{g} \sin\left(x + \frac{i}{g}\right) \tag{4}$$

where g is the grid size, the total sum increases non-linearly as a function of g. Most importantly, the total sum is independent of input value, x. This makes finding the appropriate grid weight scaling inconsistent across types of inputs and grid dimension. We present an alternative strategy, in which grid weights are initialized as:

$$\sum_{i=1}^{g} \sin\left(x + \frac{i}{g+1}\right) \tag{5}$$

In the case where frequencies are all fixed at the same constant value, the sum converges to:

$$\sum_{i=1}^{g} \sin\left(x + \frac{i}{g+1}\right) = C(g) \sin\left(x + \frac{1}{2}\right) \tag{6}$$

where C(g) is a constant that scales with g. This means that, for fixed frequencies, the scaling behavior of the model output would be independent of x.

Furthermore, we find that introducing an additional input phase term along the axis of the number of input features with values ranging from zero to π leads to stronger model performance.

Finally, to stabilize the model scaling across different grid sizes, we find a functional form that helps scale the total sum across the grid dimension as a ratio of phase terms:

$$R = Ag^{-K} + C\phi_{g+1} = \phi_g R(g) \tag{7}$$

where $A = 0.97241$, $K = 0.988440$, and $C = 0.999450$, R is a scale factor by which all phase terms are multiplied as you increase from a grid size of one upward, and ϕ_g is the phase at a particular grid size. To determine A, K, and C we perform least squares minimization of:

$$L(g,x) = \sigma^2 \left(\frac{f(g+1,x)}{f(g,x)}\right) + \left(1 - \mu \left(\frac{f(g+1,x)}{f(g,x)}\right)\right)^2 \tag{8}$$

where L is a cost function, f(g,x) is the sum of sines across input values from $-\pi$ to π, μ is the mean value and σ^2 is the standard deviation.
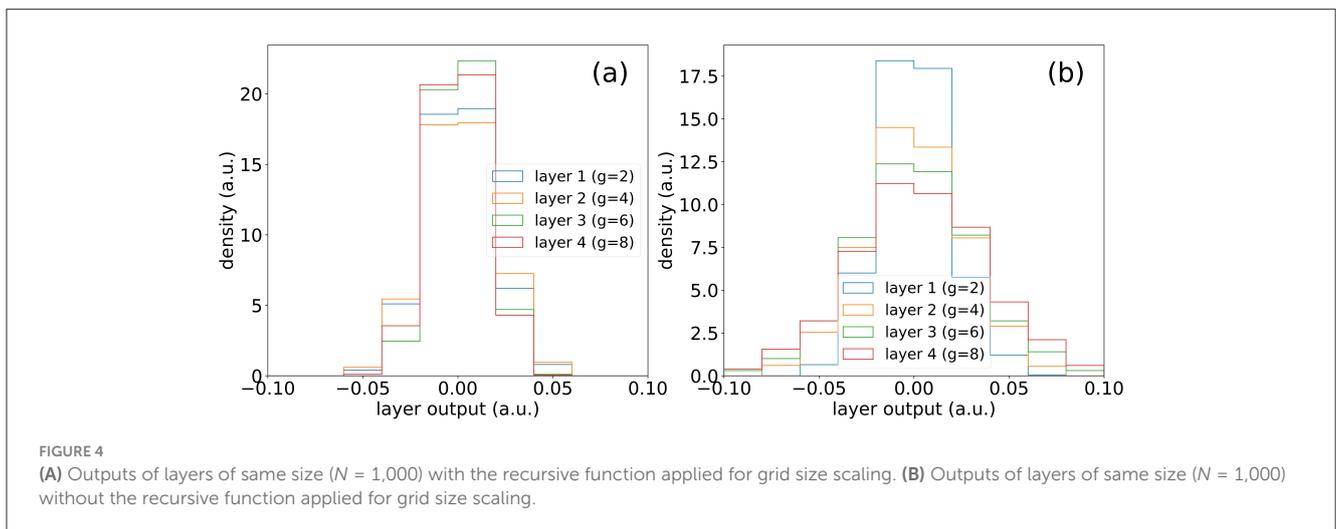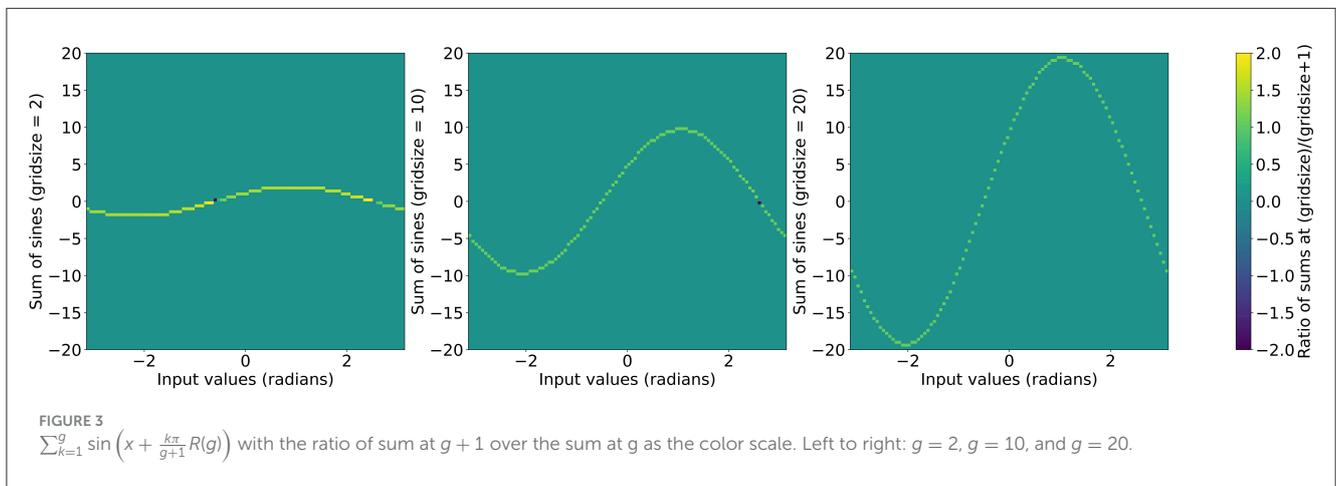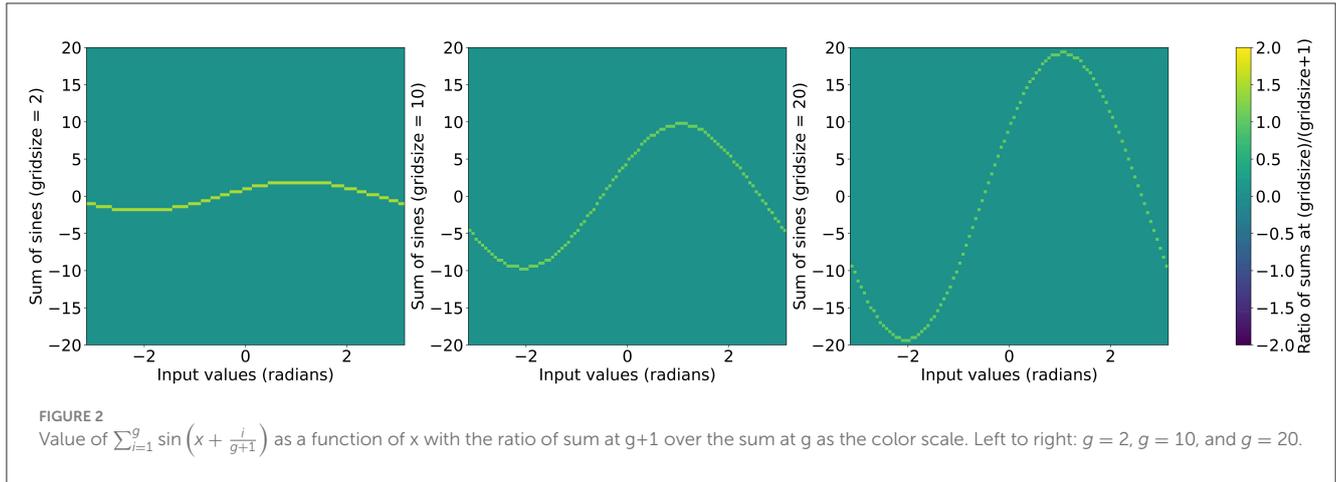
Using this functional form we can initialize layers with arbitrary grid sizes by using the exponential expression in a recursive formula while initializing the grid phase weights. The original ratios of sums of sines are shown in Figure 2 and the ratios after applying the recursive formula are shown in Figure 3.

In Figure 4B, we see the outputs of subsequently connected layers when recursive grid size phase scaling is not applied. In Figure 4A, we see the same scenario but with recursive grid size phase scaling applied and see an increase in similarity of layer outputs across various grid sizes.

3.3 Scaling of phase terms with grid size

We find a weight initialization strategy which results in strong performance and stability in higher depth models. For the first layer, the weights are initialized as a random distribution with a mean of 0 and and a standard deviation of 0.4 and for the subsequent layers, the weights are initialized from a uniform distribution between –1 and 1. This not only leads to consistent layer weight distributions, but also leads to consistent output across connected layers of same size, as shown in Figure 5B.

It also features a desirable property that, given a fixed initialization on the first layer and a random input vector, the features span a broader range of values at deeper layers, as shown in Figure 5A. This implies that no value collapse is introduced. Comparatively, we see in Figure 5C, that the model layer outputs in B-SplineKAN implementations decrease in multi-layer models, which can play a significant role in results in Section 4.2.

FIGURE 2
Value of $\sum_{i=1}^{g} \sin\left(x + \frac{i}{g+1}\right)$ as a function of x with the ratio of sum at g+1 over the sum at g as the color scale. Left to right: $g = 2$, $g = 10$, and $g = 20$.

FIGURE 3
$\sum_{k=1}^{g} \sin\left(x + \frac{k\pi}{g+1} R(g)\right)$ with the ratio of sum at $g + 1$ over the sum at g as the color scale. Left to right: $g = 2$, $g = 10$, and $g = 20$.

FIGURE 4
(A) Outputs of layers of same size ($N = 1,000$) with the recursive function applied for grid size scaling. **(B)** Outputs of layers of same size ($N = 1,000$) without the recursive function applied for grid size scaling.

3.4 Universal approximation

The combination of learnable amplitudes and sinusoidal activation functions have previously been shown to be viable for implicit neural representations (Sitzmann et al., 2020). The models have been shown to be effective for applications including control systems (Origer and Izzo, 2024), medical applications (Li et al., 2024), and physics applications (Lei et al., 2022). However, these models only satisfy universal approximation on a single layer when combined with linear transformations in sufficiently wide or deep models. By introducing an additional degree of freedom in the form of learnable

FIGURE 5
(A) Outputs of consecutive layers of different sizes in a SineKAN model. **(B)** Outputs of consecutive layers of same size in a SineKAN model. **(C)** Outputs of consecutive layers of same size in a B-SplineKAN model.

FIGURE 6
Fits of SineKAN with grid size of 100 to assorted functions.
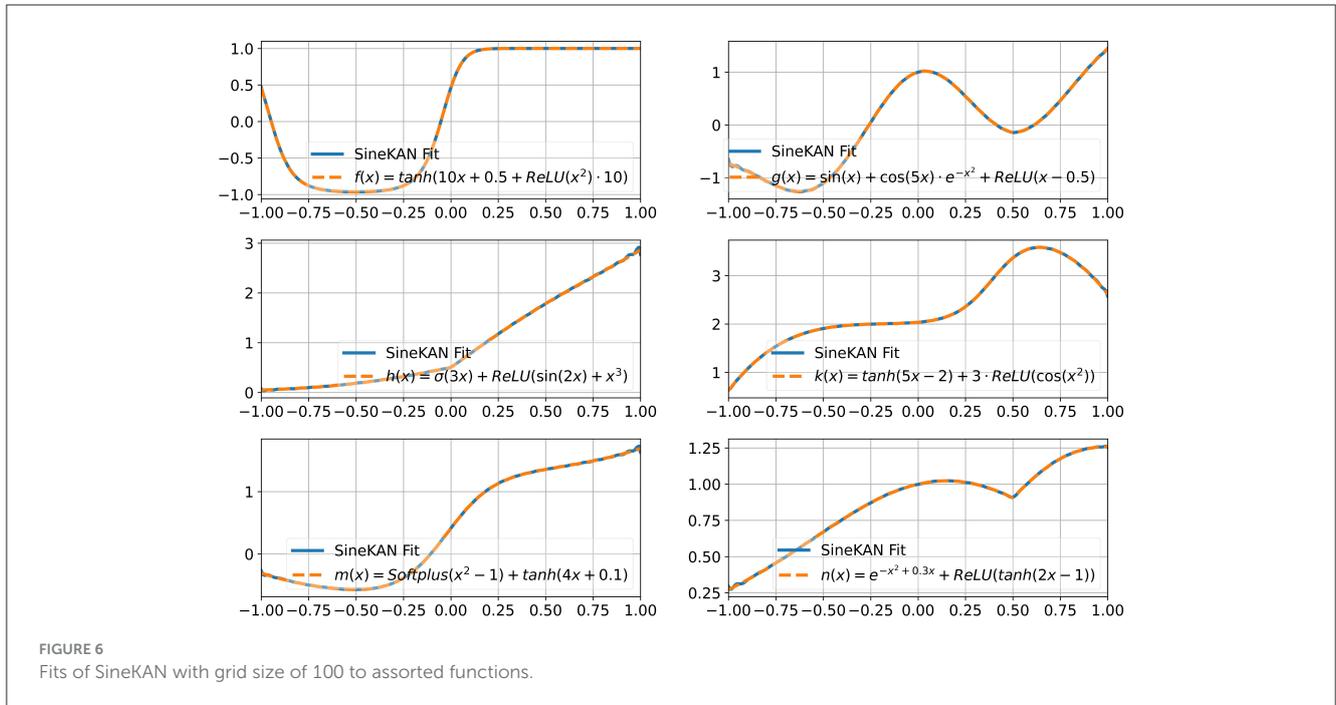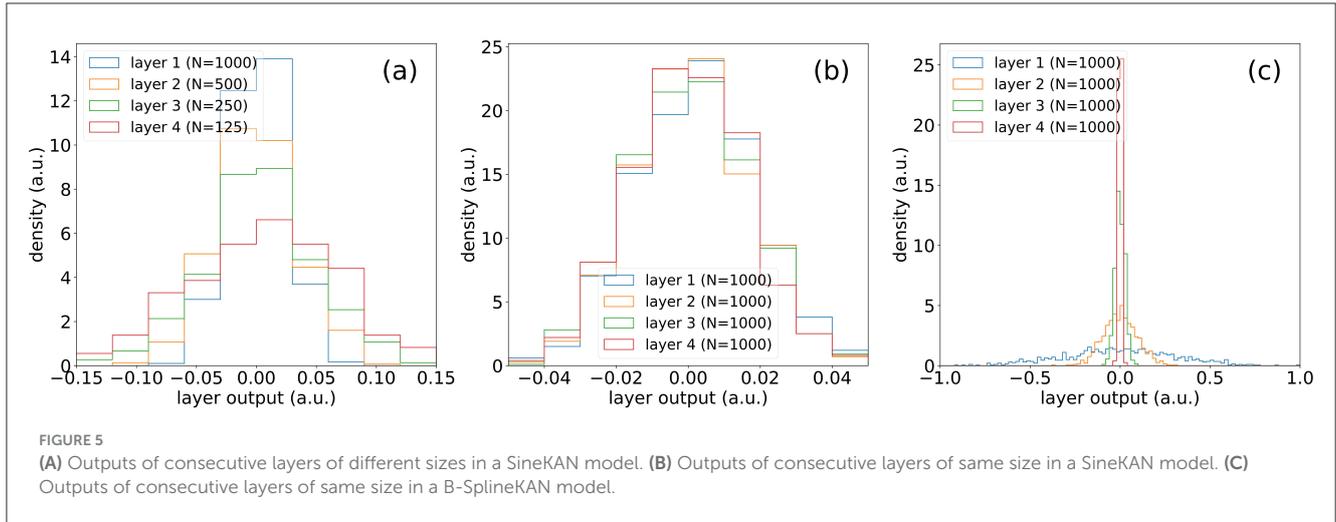
frequencies over phase-shifted grids, one can eliminate the linear transformation layers.

Any sufficiently well-behaved/smooth 1 dimensional function $f : \mathbb{R} \to \mathbb{R}$ can be expressed in terms of a *Fourier transform* $\tilde{f} : \mathbb{R} \to \mathbb{C}$ w.r.t. a continuous phase space of frequencies ω:

$$f(x) = \int_{\mathbb{R}} d\omega\, \tilde{f}(\omega) e^{i\omega x}$$

$$= \mathrm{Re}\left(\int_{\mathbb{R}} d\omega\, \tilde{f}(\omega) e^{i\omega x} \right)$$

$$= \int_{\mathbb{R}} d\omega\, A(\omega) \cos\left(\omega x + \phi'(\omega)\right)$$

$$= \int_{\mathbb{R}} d\omega\, A(\omega) \sin\left(\omega x + \phi(\omega)\right)$$
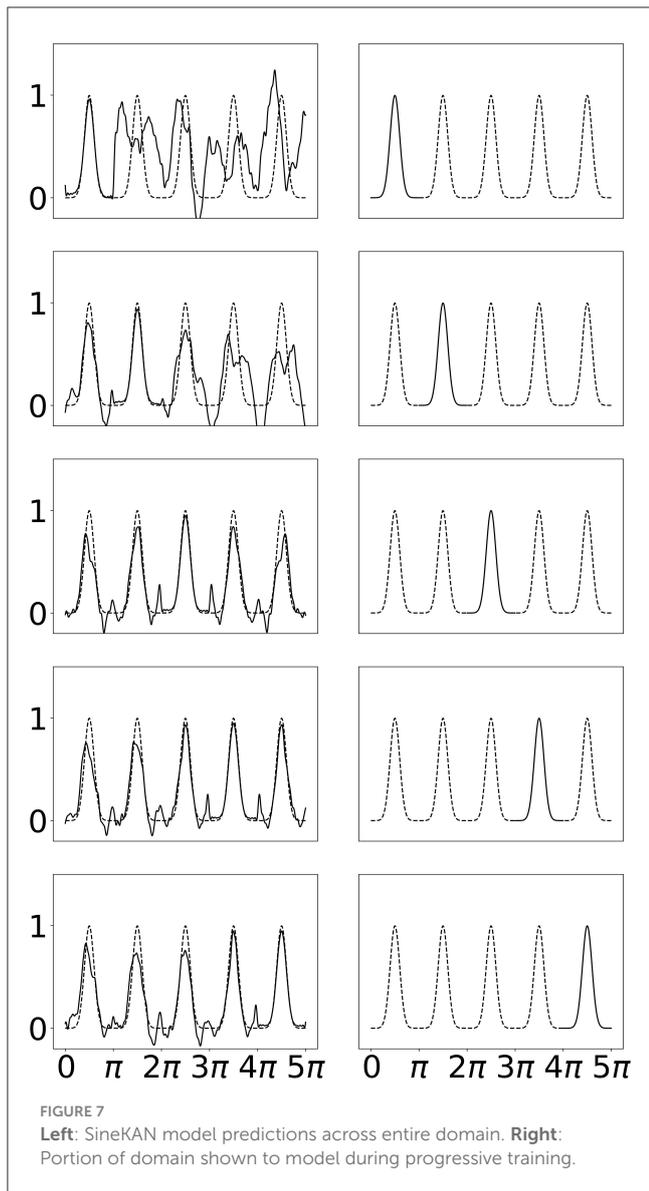
where $A(\omega)$ and $\phi(\omega) = \phi'(\omega) - \frac{\pi}{2}$ are real-valued functions. The above integral can be discretized using Riemannian sums over

a finite set of frequencies $\Omega = \{\omega_1, \omega_2 \ldots \omega_G\}$ where cardinality G of the set is the *grid size*. We henceforth propose the following variational function g_θ as an ansatz for $f(x)$:

$$g_\theta(x) = \sum_i B_i \sin\left(\omega_i x + \phi_i\right)$$

where we make the replacements $\int_{\mathbb{R}} \to \sum_i$, $d\omega A(\omega) \to B_i$ and $\omega, \phi(\omega) \to \omega_i, \phi_i$. Here, ϕ_i are G fixed, finite points from $(0, \pi + 1]$ while we treat all other subscripted symbols B_i, ω_i as weights whose values can trained to optimize a loss function between f and g_θ, which converges to the Fourier transform integral of f as $G \to \infty$. Hence, in the limit where $G \to \infty$, it's a valid candidate for a learnable activation function ansatz to be used in a Kolmogorov-Arnold Network (KAN).

In Figure 6 we show that with a layer with grid size of 100, a single-input, single-output SineKAN layer can map inputs to

FIGURE 7
Left: SineKAN model predictions across entire domain. **Right**: Portion of domain shown to model during progressive training.
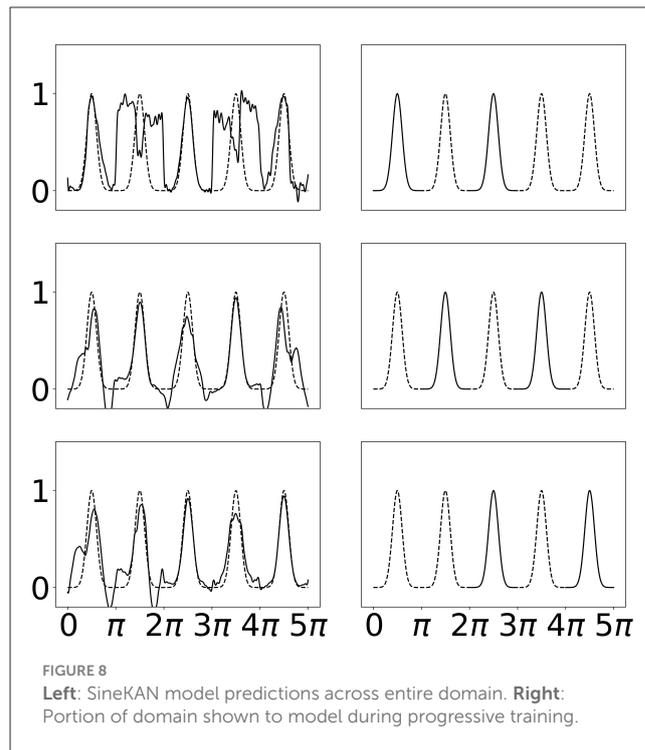
outputs in 1D functions including over non-smooth functional mappings. The functions explored are:

- $f(x) = \tanh(10x + 0.5 + \text{ReLU}(x^2) \cdot 10)$
- $g(x) = \sin(x) + \cos(5x) \cdot e^{-x^2} + \text{ReLU}(x - 0.5)$
- $h(x) = \sigma(3x) + \text{ReLU}(\sin(2x) + x^3)$
- $k(x) = \tanh(5x - 2) + 3 \cdot \text{ReLU}(\cos(x^2))$
- $m(x) = \text{Softplus}(x^2 - 1) + \tanh(4x + 0.1)$
- $n(x) = e^{-x^2 + 0.3x} + \text{ReLU}(\tanh(2x - 1))$

4 Results

4.1 Continual learning

A desirable property demonstrated for B-SplineKAN models is the ability to fit sections of the total domain without "forgetting" other sections. In Liu et al. (2024), this was performed by fitting

FIGURE 8
Left: SineKAN model predictions across entire domain. **Right**: Portion of domain shown to model during progressive training.

a repeating Gaussian waves one period at a time. The first period is fit then the first period is removed from the training data and replaced with the second period. This is repeated for each of the five periods. The original B-SplineKAN work showed that the model could learn to fit data in the new portion of the domain without catastrophically forgetting the other portions of the data it had been shown.

This task presents challenges for SineKAN due to the fact that the range of the function is periodically repeating even beyond the subsection of the domain where the model is initially fit. Due to this we expect SineKAN to have greater difficulty avoiding some amount of "forgetting". However, because the model is periodic, another behavior can emerge in which the model, if able to generalize the pattern across subsections, can potentially forecast into regions of the domain it hasn't been exposed to. This is a potentially very desirable property for tasks where there is any kind of periodic symmetry across the domain.

We see in Figure 7 that the SineKAN model exhibits the expected behavior of experiencing some amount of forgetting. We also see that by the second period it begins to generalize the periodic behavior to the third period and, by the third period, it's generally able to capture the period behavior across the entire domain. In Figure 8, we show that exposing the model to more that one disconnected period at a time (period 1 and 3 then 2 and 4 then 3 and 5) also allows the model to capture the periodic behavior across the entire domain with twice as many forward passes but with the same number of total back propagation steps. This implies that discontinuous domains wouldn't necessarily limit the model's generalization behavior.
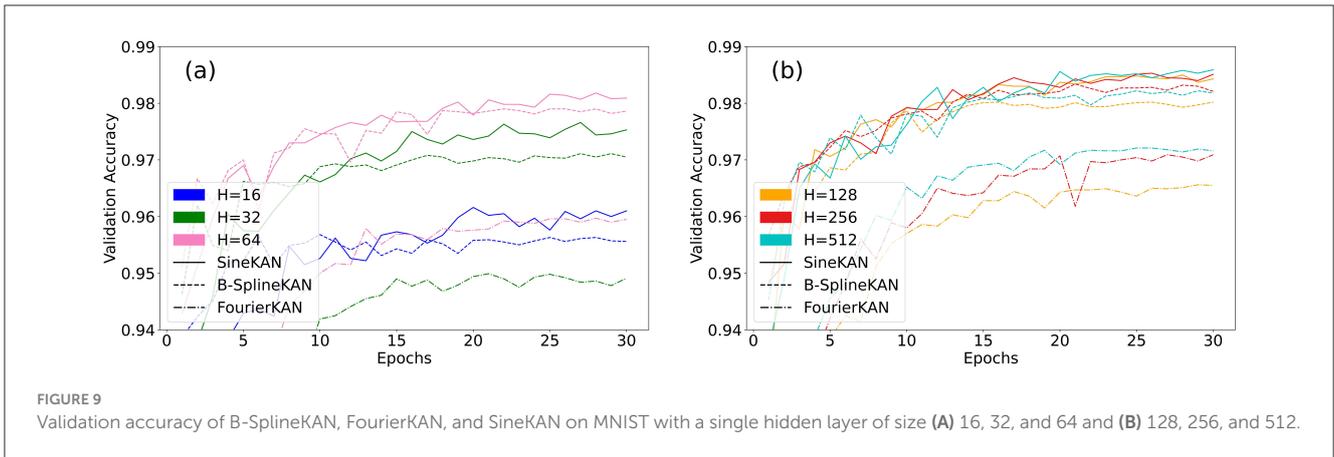
FIGURE 9
Validation accuracy of B-SplineKAN, FourierKAN, and SineKAN on MNIST with a single hidden layer of size **(A)** 16, 32, and 64 and **(B)** 128, 256, and 512.

TABLE 1 Model performance metrics by layer size with the best scores bolded.

Layer size	Model	Accuracy	Precision	Recall	F1
16	Sine	**0.9616**	**0.9617**	**0.9616**	**0.9616**
16	B-Spline	0.9568	0.9571	0.9568	0.9568
16	Fourier	0.9337	0.9337	0.9337	0.9336
32	Sine	**0.9766**	**0.9766**	**0.9766**	**0.9766**
32	B-Spline	0.9711	0.9711	0.9711	0.9711
32	Fourier	0.9499	0.9500	0.9499	0.9499
64	Sine	**0.9818**	**0.9818**	**0.9818**	**0.9818**
64	B-Spline	0.9790	0.9790	0.9790	0.9790
64	Fourier	0.9597	0.9597	0.9597	0.9596
128	Sine	**0.9850**	**0.9850**	**0.9850**	**0.9850**
128	B-Spline	0.9802	0.9802	0.9802	0.9802
128	Fourier	0.9656	0.9656	0.9656	0.9656
256	Sine	**0.9853**	**0.9853**	**0.9853**	**0.9853**
256	B-Spline	0.9834	0.9834	0.9834	0.9834
256	Fourier	0.9709	0.9709	0.9709	0.9709

FIGURE 10
B-SplineKAN, FourierKAN, and SineKAN validation accuracy on MNIST with a 1, 2, 3, and 4 hidden layers of size 128.

4.2 KAN numerical performance on MNIST

The MNIST dataset is a classification dataset which contains 60,000 training examples and 10,000 testing examples of handwritten characters (Deng, 2012). The characters can have values between 0 and 9. We train and compare single-layer B-SplineKAN and SineKAN networks on the MNIST dataset. We use a batch size 128, learning decay rate of 0.9 and learning rates of 5e-3 for B-Spline, 1e-4 for FourierKAN, and 4e-4 for SineKAN, optimized with grid search. The models are trained using the AdamW optimizer with a weight decay of 0.01 for B-SplineKAN, 1 for FourierKAN, and 0.5 for SineKAN also found via grid search (Loshchilov and Hutter, 2019). We test model performance with single hidden layer dimensions of 16, 32, 64, 128, 256, 512 training for 30 epochs using cross entropy loss (Rumelhart et al., 1986a).
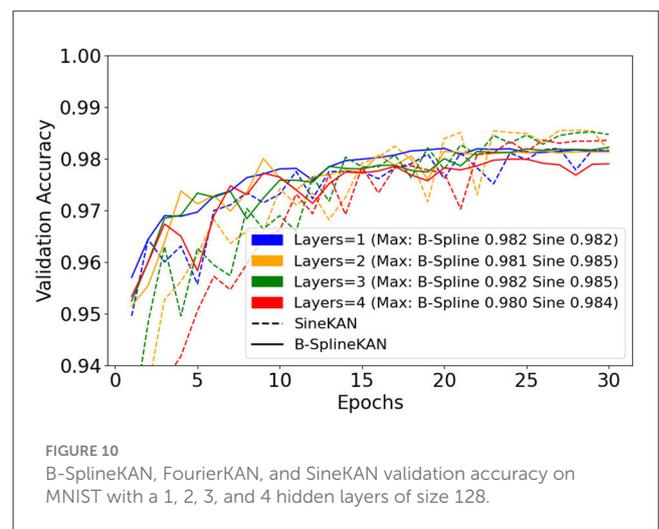
Figure 9 shows the model validation accuracy as a function of the number of epochs. The best accuracies are shown in Table 1. The SineKAN model achieves better results than the FourierKAN and B-SplineKAN models for all hidden layer sizes.

We additionally explore fitting using the same hyperparameters but with 1, 2, 3, and 4 hidden layers of size 128. Figure 10 shows that the SineKAN outperforms the FourierKAN and B-SplineKAN at lower layer depths. We also find, however, that without additional tuning of hyperparameters, all three models generally decrease in performance at higher depths.

4.3 KAN inference speeds

We benchmark the speed of SineKan and B-SplineKAN models using NVIDIA T4 GPU with 16GB of RAM. We test performance on variable batch sizes of 16, 32, 64, 128, 256, and 512 on single inputs of 784 features using a single hidden layer of size 128 with a grid size of 8. We test performance on single hidden layer hidden dimensions of 16, 32, 64, 128, 256, and 512 under the same
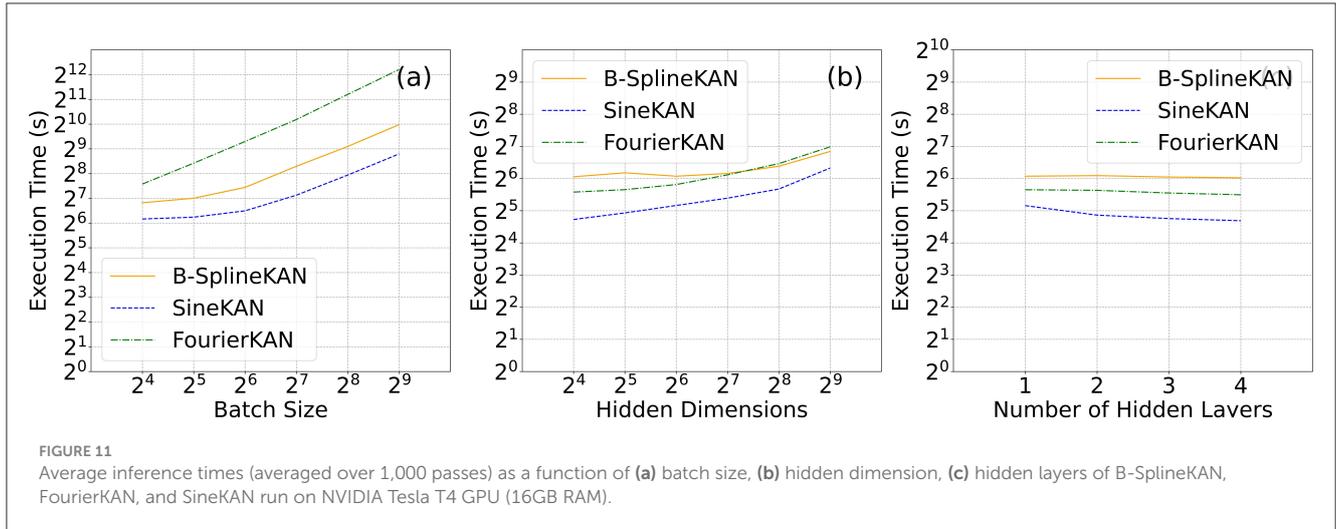
FIGURE 11
Average inference times (averaged over 1,000 passes) as a function of **(a)** batch size, **(b)** hidden dimension, **(c)** hidden layers of B-SplineKAN, FourierKAN, and SineKAN run on NVIDIA Tesla T4 GPU (16GB RAM).

TABLE 2 MLP performance metrics by layer size.

Layer size	Model	Accuracy	Precision	Recall	F1
16	MLP	0.9183	0.9181	0.9183	0.9181
32	MLP	0.9557	0.9557	0.9557	0.9556
64	MLP	0.9682	0.9682	0.9682	0.9682
128	MLP	0.9797	0.9797	0.9797	0.9797
256	MLP	0.9822	0.9822	0.9822	0.9822
512	MLP	0.9842	0.9842	0.9842	0.9842
1024	MLP	0.9843	0.9843	0.9843	0.9843
2048	MLP	0.9863	0.9863	0.9863	0.9863

conditions. We test performance with single batch of 784 features with 1, 2, 3, and 4 hidden layers of size 128.

As Figure 11 shows, the SineKAN has the best inference times at all batch sizes compared to B-SplineKAN and FourierKAN. It also has the best inference times across all hidden layer dimensions explored with all three models showing roughly flat scaling as a function of model depth. Due to differences in hardware optimization and cuda kernel optimization, it is difficult to directly compare the performance of the three models on-device. To account for idealized performance optimizations, we also derive analytically the expected model scaling behavior in Appendix. We find that the expected scaling for the three models in approximate FLOP compute units are as follows:

- SineKAN: $\mathcal{O}(2b\, d_{out}\, d_{in}\, g)$
- FourierKAN: $\mathcal{O}(4b\, d_{out}\, d_{in}\, g)$
- B-SplineKAN: $\mathcal{O}(2b\, d_{out}\, d_{in}\, (g+s))$

Here b is the batch size, d_{in} and d_{out} are the input and output dimensions, g is the grid size, and s is the basis-spline order. We therefore expect that, with full device-level and software-level optimization, the relative performance of B-SplineKAN will scale like $g/(g+s)$ relative to SineKAN and FourierKAN to scale like half the speed of SineKAN.

4.4 MLP comparison

We also compare performance to MLP in Table 2. For MLP we performed a similar grid search and found a learning rate of 8e-4, weight decay of 0.01, and learning decay rate of 0.9 to be the best performing. We find that MLP exceeds SineKAN's performance at a grid size of 8 and hidden layer size of 256 once it reaches a hidden layer size of 2,048.

Due to previously mentioned differences in device-level and software-level optimizations for the different models, it's difficult to directly compare the performance of MLP and SineKAN. However, the characteristic FLOPs of an MLP layer scales as $\mathcal{O}\left(2bd_{out}d_{in}\right)$. We therefore would consider that, for a SineKAN model with a grid size of 8, a competitive MLP performance would be at 8 times the hidden layer dimension. However, it's also worth acknowledging that as the first hidden layer size in the MLP increases, additional parameters are added in the output layer. MLP under-performing SineKAN (0.9853 accuracy) at a hidden layer size of 1,024 (0.9843 accuracy) and outperforming SineKAN at a hidden layer size of 2,048 (0.9863 accuracy) is consistent with a competitive performance to MLP as a function of FLOPs under full optimization.

5 Discussion

The SineKAN model, which uses sine functions as an alternative to existing baseline B-Spline activation functions (Liu et al., 2024), shows very good performance on the benchmark task. Model stabilization techniques described in Section 3.2 lead to consistent model layer output weights at different depths and sizes of phase shift grids. We also show that it actively mitigates value collapse in deep models.

In Section 4.2, we show that SineKAN increasingly outperforms the B-SplineKAN model at very large hidden layer dimensions. These results suggest that the SineKAN model may be a better choice compared to B-SplineKAN for scalable, high-depth, large batch models such as large language models (LLMs). However, when comparing to MLP we also account for difference in scaling

resulting from lack of a grid. We find that MLP models perform similarly to SineKAN when accounting for idealized model scaling. However, at very large sizes (hidden dimension of 2048), MLP outperforms SineKAN with comparable idealized inference time.

We show in Section 4.3 that the SineKAN model is faster than both FourierKAN and B-SplineKAN as a function of batch size, hidden layer dimension, and depth. Due to device-level and software-level optimizations to computation, we also derive empirically in Appendix what the expected true scaling is under optimal conditions. We find that SineKAN is roughly (g+s)/g times faster than B-SplineKAN where g is grid size and s is basis-spline order and roughly two times faster than FourierKAN.

We also found that SineKAN had a significantly different optimal learning rate and weight decay compared to B-SplineKAN and FourierKAN which motivates the idea that a fair comparison cannot be done across different KAN implementations without performing a grid search to find optimal hyperparameters. Further, we also showed in Section 3.3, that B-SplineKAN has an inherent flaw in scaling to multi-layer models in that it increasingly constricts layer outputs at higher depths. This likely necessitates additional layer normalization between B-SplineKAN layers. We recommend that approaches for stabilizing model weights at different sizes and depths, similar to those outlined in Sections 3.2, 3.3, should be employed in other KAN models to improve deep model stability and performance.

Regarding general KAN properties, SineKAN is able to demonstrate some avoidance of catastrophic forgetting in the continual learning task. It also shows a potentially favorable behavior in generalization of repeating patterns to as-of-yet unseen portions of the domain space. However, we introduce a recursive phase scaling which improves model stability and performance at different model sizes. This makes the current implementation of SineKAN incapable of directly transferring weights to a larger grid. Absence of grid expandability could present major limitations in use cases where extremely large grid sizes might be required. Further, SineKAN currently does not support symbolic expressions.

In summary, sinusoidal activation functions appear to be a promising candidate in the development of Kolmogorov-Arnold models. SineKAN has superior performance in inference speed and accuracy, as well as multi-layer scaling when compared with B-SplineKAN. However, a number of other activation functions mentioned in Section 1 have also shown to have superior inference speed and better numerical performance. Further exploration is needed to compare the performance both in terms of inference speed and numerical performance on the broad range of KAN implementations, and on a broader range of tasks, under fair conditions.

6 Conclusion

We present the SineKAN model, a sinusoidal activation function alternative to B-SplineKAN and FourierKAN Kolmogorov-Arnold Networks and multi-layer perceptrons. We find that SineKAN has one desirable property of KAN models in avoidance of catastrophic forgetting during continual learning and an additional property of generalization of patterns to unseen regions of the domain. We also find that this model leads to better numerical performance on the MNIST benchmark task compared

to other KAN models and comparable performance to MLP when all models are trained using near-optimal hyperparameters found with a parameter grid search. The SineKAN model outperforms B-SplineKAN at higher hidden dimension sizes with more predictable performance scaling at higher depths. We further find that SineKAN outperforms efficient implementations of FourierKAN and B-SplineKAN on speed benchmarks and are expected to outperform even at full device- and software-level optimization. We find that SineKAN performs similarly to MLP when accounting for idealized scaling though MLP can still outperform SineKAN given sufficiently large hidden layer dimensions.

Future work should aim to compare other KAN models under similar, optimized conditions. Additional explorations are also needed regarding deep-model stabilization techniques for various KAN models. Due to competitive performance of SineKAN models with MLP, we find it worth exploring use of SineKAN models in place of MLP in more complex architectures involving feature extractors such as convolutional neural networks (He et al., 2015) and transformers (Vaswani et al., 2017). Further work is also needed to include features in SineKAN models which are available in some other KAN models (Liu et al., 2024) such as symbolic equation representing and transfer of weights during grid size expansion of the model. Finally, further exploration is needed to determine use cases which best leverage the periodic behavior of the SineKAN model.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

ER: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. DR: Writing – review & editing, Methodology, Conceptualization, Validation, Investigation, Software. SG: Funding acquisition, Project administration, Resources, Supervision, Writing – original draft, Writing – review & editing, Methodology.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Aghaei, A. A. (2024a). fKAN: fractional Kolmogorov-Arnold networks with trainable Jacobi basis functions. *arXiv preprint arXiv:2406.07456*.

Aghaei, A. A. (2024b). rKAN: rational Kolmogorov-Arnold networks. *arXiv preprint arXiv:2406.14495*.

Bozorgasl, Z., and Chen, H. (2024). WAV-KAN: wavelet Kolmogorov-Arnold networks. *arXiv preprint arXiv:2405.12832*.

Braun, J., and Griebel, M. (2009). On a constructive proof of Kolmogorov's superposition theorem. *Constr. Approxim.* 30, 653–675. doi: 10.1007/s00365-009-9054-2

Cao, H. (2024). *Efficient-kan*. Available at: https://github.com/Blealtan/efficient-kan (accessed November 3, 2024).

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Process. Mag.* 29, 141–142. doi: 10.1109/MSP.2012.2211477

Gallant, R., and White, H. (1988). "There exists a neural network that does not make avoidable mistakes," in *IEEE International Conference on Neural Networks*, 657–664. doi: 10.1109/ICNN.1988.23903

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. doi: 10.1109/CVPR.2016.90

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 359–366. doi: 10.1016/0893-6080(89)90020-8

Kolmogorov, A. N. (1956). On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables. *Doklady Akademii Nauk* 108, 383–387.

Kolmogorov, A. N. (1957). "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," in *Doklady Akademii Nauk* (Russian Academy of Sciences), 953–956.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1, 541–551. doi: 10.1162/neco.1989.1.4.541

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lei, M., Tsang, K. V., Gasiorowski, S., Li, C., Nashed, Y., Petrillo, G., et al. (2022). Implicit neural representation as a differentiable surrogate for photon propagation in a monolithic neutrino detector. *arXiv preprint arXiv:2211.01505*.

Li, Y., Lou, A., Xu, Z., Wang, S., and Chang, C. (2024). Leveraging sinusoidal representation networks to predict fMRI signals from EEG. *arXiv preprint arXiv:2311.04234*.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., et al. (2024). Kan: Kolmogorov-Arnold networks. *arXiv preprint arXiv:2404.19756*.

Loshchilov, I., and Hutter, F. (2019). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Origer, S., and Izzo, D. (2024). Guidance and control networks with periodic activation functions. *arXiv preprint arXiv:2405.18084*.

Parascandolo, G., Huttunen, H., and Virtanen, T. (2016). "Taming the waves: Sine as activation function in deep neural networks," in *ICLR 2017*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning representations by back-propagating errors. *Nature* 323, 533–536. doi: 10.1038/323533a0

Rumelhart, D. E., McClelland, J. L., and PDP Research Group (1986b). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*. Cambridge, MA, USA: MIT Press.

Shukla, K., Toscano, J. D., Wang, Z., Zou, Z., and Karniadakis, G. E. (2024). A comprehensive and fair comparison between MLP and KAN representations for differential equations and operator networks. *arXiv preprint arXiv:2406.02917*.

Sidharth, S. S., Keerthana, A. R., and Anas, K. P. (2024). Chebyshev polynomial-based Kolmogorov-Arnold networks: an efficient architecture for nonlinear function approximation. *arXiv preprint arXiv:2405.07200*.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. (2020). "Implicit neural representations with periodic activation functions," in *Advances in Neural Information Processing Systems*, 7462–7473.

Sopena, J. M., Romero, E., and Alquezar, R. (1999). "Neural networks with periodic and monotonic activation functions: a comparative study in classification problems," in *Proceedings of the ICANN*. doi: 10.1049/cp:19991129

Ta, H.-T. (2024). BSRBF-KAN: a combination of b-splines and radial basic functions in Kolmogorov-Arnold networks. *arXiv preprint arXiv:2406.11173*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *CoRR, abs/1706.03762*.

Wong, K. W., Leung, C. S., and Chang, S. J. (2002). "Handwritten digit recognition using multilayer feedforward neural networks with periodic and monotonic activation functions," in *Object Recognition Supported by User Interaction for Service Robots* (IEEE), 106–109.

Xu, J., Chen, Z., Li, J., Yang, S., Wang, W., Hu, X., et al. (2024). Fourierkan-GCF: fourier Kolmogorov-Arnold network-an effective and efficient feature transformation for graph collaborative filtering. *arXiv preprint arXiv:2406.01034*.

Zhumekenov, A., Uteuliyeva, M., Kabdolov, O., Takhanov, R., Assylbekov, Z., and Castro, A. J. (2019). Fourier neural networks: a comparative study. *arXiv preprint arXiv:1902.03011*.

Appendix

Code

The SineKAN code can be found at https://github.com/ereinha/SineKAN.

Model scaling derivations

For these derivations we assume that addition, multiplication, and subtraction will require on average 1 FLOP, division and exponential will require on average 5 FLOPs, and trigonometric functions (sine/cosine) will require on average 10 FLOPs. We will also assume any boolean logic and reshaping will require 0 FLOPs. Here b is the batch size, g is the grid size, s is the basis-spline order, d_{in} is the input dimension, d_{out} is the output dimension.

SineKAN Layer:
Initial multiplication:

$$M_1 = bd_{in}g \tag{A1}$$

Add phase:

$$A_1 = bd_{in}g \tag{A2}$$

Trigonometric evaluations:

$$N_{sin} = 10bd_{in}g \tag{A3}$$

Einsum multiplications:

$$M_2 = bd_{out}d_{in}g \tag{A4}$$

Einsum additions:

$$A_2 = bd_{out}(d_{in}g - 1) \tag{A5}$$

Add bias:

$$A_3 = bd_{out} \tag{A6}$$

Total FLOPs: $bd_{in}g(2d_{out} + 12) + bd_{out}$
Leading order: $O(2bd_{in}gd_{out})$

FourierKAN Layer:
Initial multiplication:

$$M_1 = bd_{in}g \tag{A7}$$

Trigonometric evaluations:

$$N_{cos} = N_{sin} = 10bd_{in}g \tag{A8}$$

Multiplication by Fourier coefficients:

$$M_2 = 2bd_{out}d_{in}g \tag{A9}$$

Addition over dimensions:

$$A_1 = 2bd_{out}(d_{in}g - 1) \tag{A10}$$

Cosine and sine combination:

$$A_2 = bd_{out} \tag{A11}$$

Add bias:

$$A_3 = bd_{out} \tag{A12}$$

Total FLOPs: $bd_{in}g(4d_{out} + 21) + bd_{out}$
Leading order: $O(4bd_{in}gd_{out})$

EfficientKAN Layer:
SiLU activation:

$$N_{silu} = 13bd_{in} \tag{A13}$$

Base linear multiplication:

$$M_{base} = bd_{out}d_{in} \tag{A14}$$

Base linear addition:

$$A_{base} = bd_{out}(d_{in} - 1) \tag{A15}$$

B-Spline basis calculation:

$$N_{spline} = 17sbd_{in}(g + 2s) \tag{A16}$$

Spline linear multiplication:

$$M_{spline} = bd_{out}d_{in}(g + s) \tag{A17}$$

Spline linear addition:

$$M_{spline} = bd_{out}[d_{in}(g + s) - 1] \tag{A18}$$

Combine outputs:

$$A_{combine} = bd_{out} \tag{A19}$$

Total FLOPs: $13bd_{in} + 2bd_{out}d_{in} + 17sbd_{in}(g + 2s) + 2bd_{out}d_{in}(g + s)) + bd_{out}$
Leading order: $O(2bd_{out}d_{in}(g + s))$